электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

 # 01, январь 2016

УДК 004.021

Разработка метода обнаружения тупика на модели параллельных процессов формализованных сетью Петри

Омарова М.О., *студент*
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Программное обеспечение ЭВМ и информационные технологии»

Научный руководитель: Рудаков И.В., к.т.н., доцент
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Программное обеспечение ЭВМ и информационные технологии»
irudakov@bmstu.ru

Введение

Современный этап развития вычислительной техники характеризуется интенсификацией использования технических, информационных и программных средств, сосредоточенных в различных вычислительных системах. Одно из явлений, нарушающих нормальную работу и возникающих при мультидоступе к общим ресурсам, заключается в том, что некоторые процессы могут оказаться в, так называемой, тупиковой ситуации. Тупиковой называется такая ситуация, когда два или более процессов не могут выполняться без принятия чрезвычайных системных мер из-за взаимного ожидания высвобождения ресурсов. Причины тупиковых ситуации кроются в недостаточном совершенстве механизмов управления. Они могут содержаться в математическом обеспечении вычислительной системы или программах пользователей. Вычислительные системы и сети, в которых происходят неустраняемые тупиковые ситуации, чрезвычайно нежелательны для пользователей ввиду резкого снижения пропускной способности. Поэтому необходимо иметь протоколы и механизмы управления, которые не допускают тупиковых ситуаций, а также средства автоматического обнаружения и быстрого устранения тупиков, если они возникают.

Метод анализа сети Петри на соответствие представлению параллельных процессов с конкуренцией

Тупиковая ситуация может возникнуть при разделении ресурса параллельными

процессами, поэтому первым вопросом предъявляемым к сети представляющую некую вычислительную систему является существование параллелизм и конкуренции между процессами, т. е. содержится ли в сети параллельный процесс с конкуренцией. Процесс называется параллельным с конкуренцией если любая пара его различных элементов связана одним из отношений: следствия — *li*, *параллелизма — со и конкуренции — con*.
Определим эти отношения между элементами:

- Отношение следствия *li* между элементами процесса определяется следующим образом: $x\ li\ y \leftrightarrow (x < y\ or\ y < x) \vee (x \equiv y)$, отношение означает, что возможен только один из двух вариантов либо *x* реализуется раньше чем *y*, либо *y* реализуется раньше чем *x*.
- Отношение параллелизма *co* между элементами процесса определяется следующим образом: $x\ co\ y \leftrightarrow (\ \neg(x < y) \wedge \neg(y < x)) \vee (x \equiv y)$, данное отношение не вводит никаких ограничений на порядок следования элементов и не устанавливают никаких причинно-следственных связей между ними.
- Отношение конкуренции con между элементами процесса определяется следующим образом: $x\ con\ y \leftrightarrow (\ x < y \wedge y < x) \vee (x \equiv y)$, данное отношение позволяет элементам реализоваться в любом порядке.

Для того что бы сеть Петри представляла собой сеть описывающую параллельные процессы необходимо ввести следующие ограничения:

1. Для $X = P\ U\ T$ и для любого *x,y* из *X*, справедливо следующее условие $x\ F^*\ y \rightarrow \neg(y\ F^*\ x)$, если $x \neq y$, т. е. сеть не содержит циклов (T — непустое множество переходов; P — непустое множество мест; F — функция инцидентности; F* - транзитивное замыкание F).
2. $H(N) \neq 0$ и для любого *x* из X : $D^{-1}(x)$ — конечен, т. е. имеется место без входного перехода и не содержит бесконечного обратного пути (влево) (H(N) — множество головных мест не имеющих входных переходов; D(x) — конечный или бесконечный путь начинающийся с места *x;* $D^{-1}(x)$ — конечный или бесконечный обратный путь (влево) от места *x*).
3. Для любого t из T верно условие $'t \neq 0$ и $t' \neq 0$, любой переход должен иметь хотя бы одно входное и выходное место.
4. Для любого p из P выполняется условие $M_0(p)$ = { 1 - если p из H(N), 0 - в остальных случаях} — начальная разметка такая, что только головные места содержат по одной фишке
5. Каждое место в сети имеет не больше одного входного и одного выходного

перехода, т. е. для любого *p* из *P* выполняется условие: |'p| ≤ 1 ∧ |*p'*| ≤ *1*.

Сети Петри предназначенные для описания параллельных процессов с конкуренцией, строятся на основе сетей процессов с добавлением в них специальных мест, называемых ресурсами, и дуг, связывающих эти места с переходами особым способом.

На сеть с конкуренцией вводятся дополнительные ограничения:

6. Для любого *p* из *Pcon* выполняется условие $M_0(p) = 1$ — места-ресурсы имеют единичную разметку. (Pcon — множество мест-ресурсов которые разделяют переходы, то есть из которых выходят два или более перехода).

7. Для любого *p* из *Pcon* выполняется условие: |*'p*| = |*p'*| ≥ *2* — в места-ресурсы входит столько же дуг сколько и выходит, но минимум два.

8. для любого выходящего из места-ресурса перехода t2 существует переход t1, что найдется интервал I(t1, t2) и наоборот для любого входящего в места-ресурса перехода t1 существует переход t2, что найдется - I(t1, t2).

Метод определения ацикличности сети Петри

Существует множество алгоритмов обнаруживающих циклы в направленных графах, рассмотрим простой алгоритм с возвратами:

1. Для каждого головного места ph из H(N) выполняются следующие пять шагов

2. Задаем начальные условия: L- пустой список, все ребра не маркированы

3. Текущий узел добавляем в конец списка L и проверяем количество появлений узла в списке. Если узел содержится в двух местах, то граф содержит цикл и работа алгоритма завершается

4. Для заданного узла смотрим, выходит ли из него хотя бы одно ребро не содержащееся в списке возвратов — B и не являющееся местом-ресурсом. Если да, то переходим к шагу 5, иначе переходим к шагу 6

5. Случайным образом выбираем любое не маркированное исходящее ребро и отмечаем его. Переходим по нему к новому текущему узлу и возвращаемся к шагу 3

6. Теперь мы зашли в тупик. Удаляем последний узел из списка и помещаем его в список возвратов — B, возвращаемся к предидущему узлу, то есть к тому что был текущим перед тупиковым узлом. Обозначаем его текущим узлом и возвращаемся к шагу 3. Если это первоначальный узел, то граф не содержит циклов и алгоритм завершает работу.

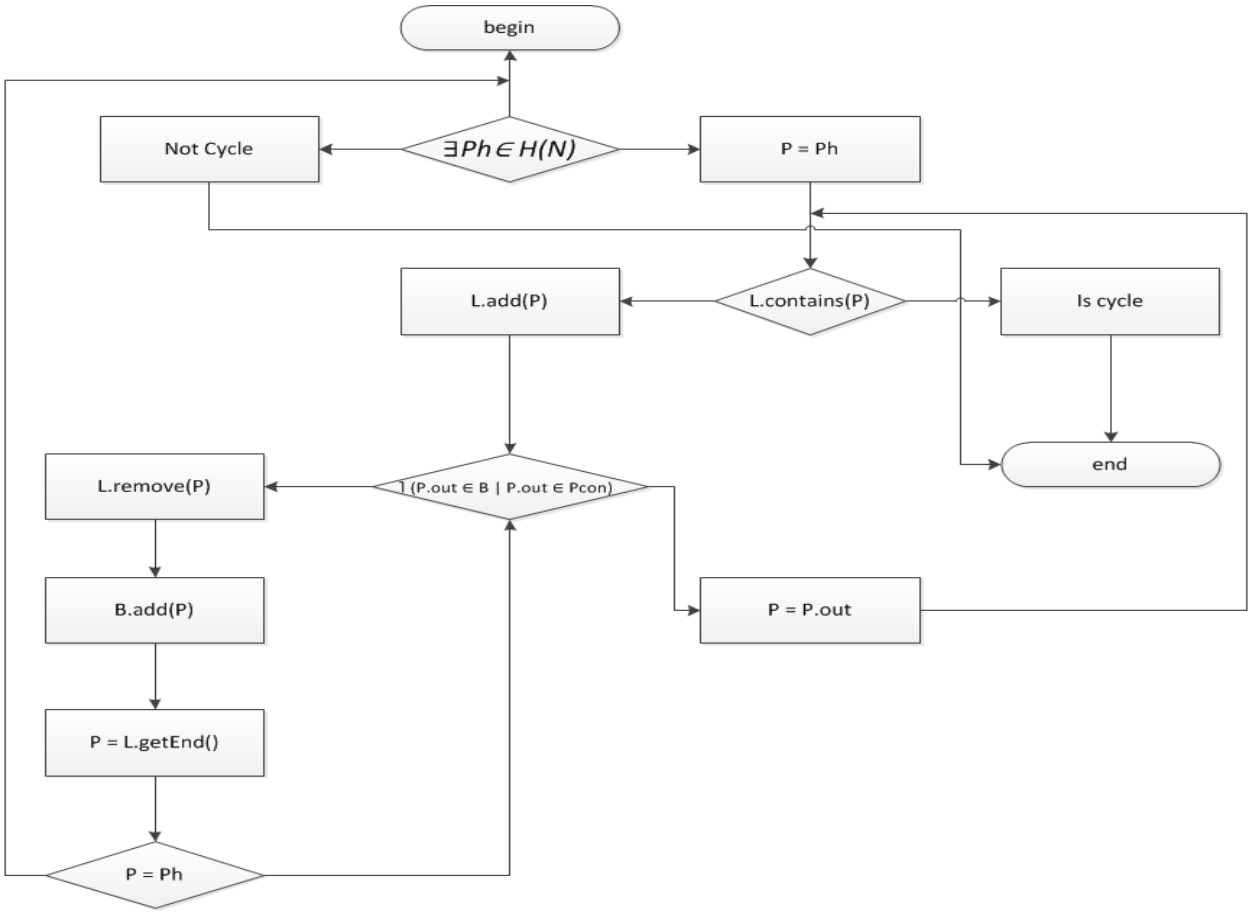

Рис.1. Алгоритм определения ацикличности сети

Метод определения К-плотности сети Петри

Для того что бы формальное представление сети семантически совпадало с понятием параллельного процесса необходимо ввести дополнительное условие К-плотности сети. К-плотной называется сеть в которой любое li-сечение (максимальные линии, которые не содержатся ни в каких других линиях, где линия это множество элементов связанных отношением предшествования) и любое со-сечение (максимальные разрезы, которые не содержатся ни в каких других разрезах, где разрез это множество элементов связанных отношением параллелизма) пересекаются ровно по одному элементу. Любая конечная сеть параллельного процесса К-плотна.

Алгоритм определения К-плотности сети:

1. Задаем начальные условия: пустые списки Li — сечение и Co — сечение
2. Для каждого головного места *p* из *H(N)* выполняются следующие два шага
3. Добавляем текущий элемент *p* в список Li — сечение

4. Идем к следующему элементу в сети, если он не содержится в множестве позиций-ресурсов (Pcon), то добавляем его в список Li- сечение и помечаем как текущий - p, переходим к шагу 3.

5. Для каждого места-ресурса *p* из *Pcon* выполняются следующие два шага

6. Добавляем текущий элемент *p* ресурс в список Co — сечение

7. Добавляем все элементы выходящие из текущего элемента-ресурса в список Co — сечение

8. Если пересечение списков Li-сечений и Co-сечений содержит ровно один элемент, то сеть является К-плотной иначе сеть не является К-плотной.

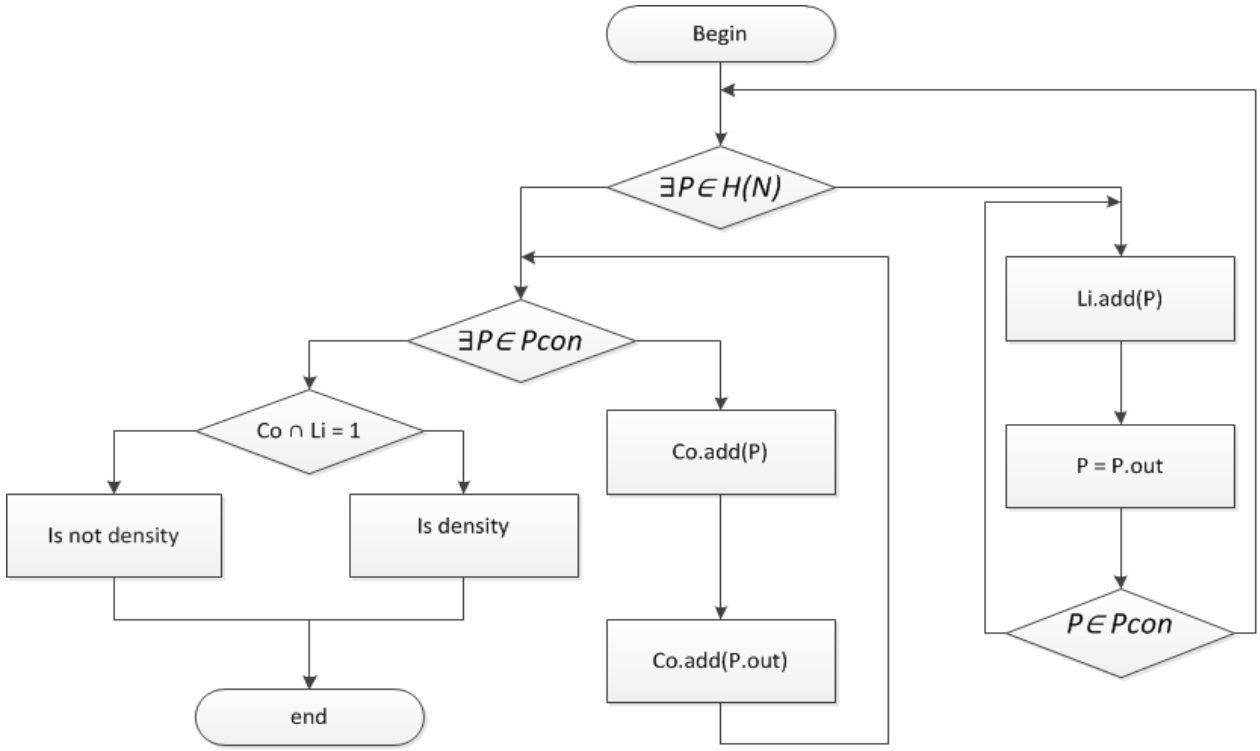

Рис. 2. Алгоритм определения К-плотности сети Петри

Сеть описывающую параллельный процесс с конкуренцией можно разделить на множество интервалов по разделяемым ресурсам, множество элементов находящиеся в отношении следования li принадлежат интервалу если начальный элемент является входным переходом для ресурса, а конец интервала выходным для этого же ресурса. Если в разных интервалах их начала являются выходными переходами для некоторого ресурса, а их концы — входными для того же ресурса, то данные интервалы называются критическими по данному ресурсу. В К-плотной сети любой интервал конечен.

Из приведенных выше условий ацикличности, ограничения начальной маркировки

и конечности сети следует что сеть является безопасной.

Метод определения тупика в случае существования одного ресурса определенного класса.

Из определения тупиковой ситуации, что множество процессов находится в тупиковой ситуации, если каждый процесс из множества ожидает ресурса, которое отдано другому процессу данного множества, следует что множество критических интервалов по ресурсам не должно содержать пересечений по этим ресурсам.

И так достаточное условие возникновения тупика, отсутствие пересечений по критическому ресурсу между множеством критических интервалов по этим ресурсам.

Алгоритм построения критических интервалов в безопасной сети процессов:

1. Задаем начальные условия: con_li — матрица, представляет собой множество критических интервалов по местам-ресурсам.
2. Берем позицию pi из множества мест ресурсов *Pcon* и помечаем ее как текущую позицию p, то есть p=pi.
3. Как текущий переход t помечаем переход из множества выходящих из текущей позиции p.
4. Помещаем в i-ю строку матрицы con_li все позиции которые входят в текущий переход t
5. Если существует позиция выходящая из текущего перехода t, не являющаяся местом-ресурсом - $\exists p_j \in t.out \mid \neg(p_j \in Pcon)$, то помещаем ее в i-ю строку матрицы con_li и помечаем как текущую позицию p, иначе убираем позицию p из списка и помечаем последнюю позицию i-ой строки матрицы con_li как текущую позицию p.
6. Если вернулись к позиции pi, т.е. pi = p, то переходим к шагу 7 иначе возвращаемся к шагу 3
7. Если имеется необработанный переход выходящий из позиции p, то переходим к шагу 3, иначе переходим к шагу 8
8. Если имеется необработанная позиция pi из множества *Pcon*, то переходим к шагу 2, иначе завершаем алгоритм

Если в i-ой строке матрицы con_li содержится j-ый ресурс, а так же в j-ой строке матрицы con_li содержится i-ый ресурс, то возможна блокировка по ресурсам pi и pj;

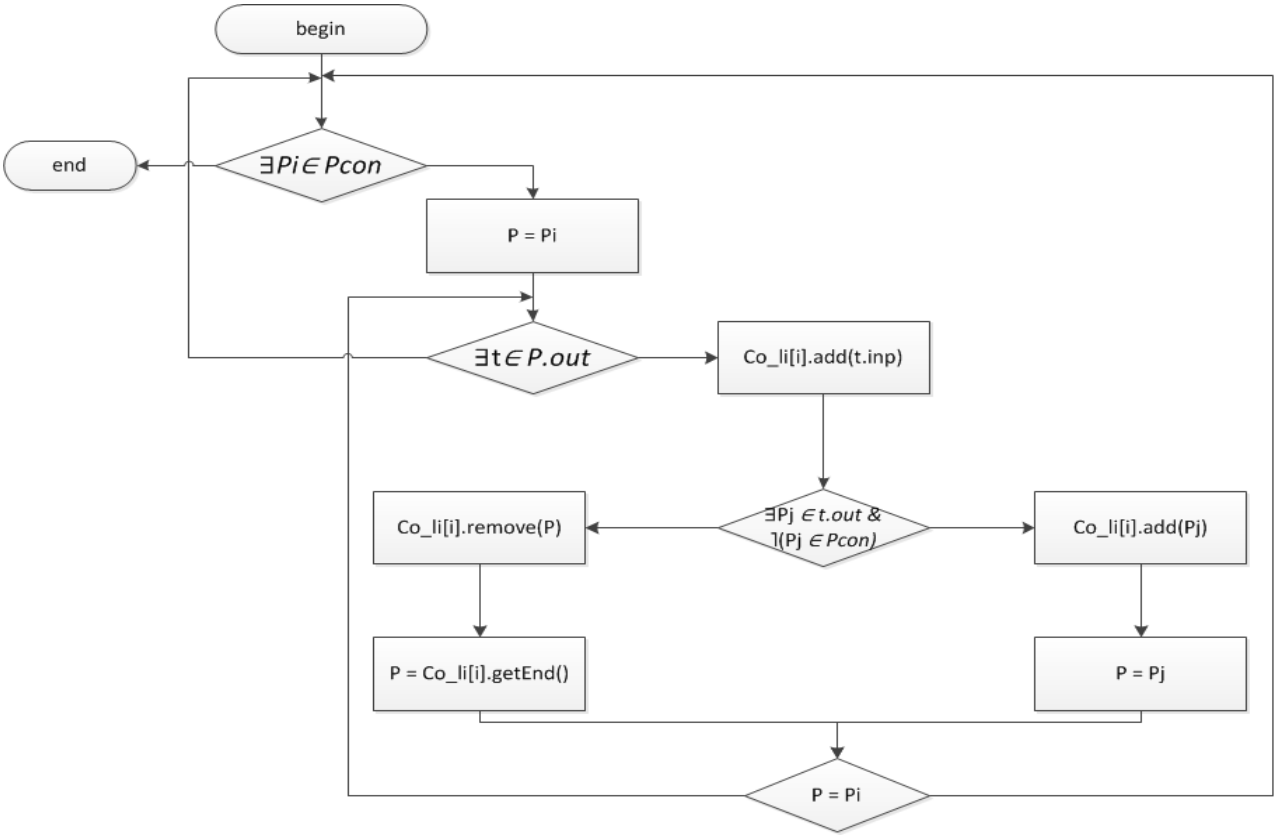

Рис. 3. Алгоритм построения критических интервалов

Метод определения тупика в случае существования множества ресурсов определенного класса

Алгоритм обнаружения блокировки при существовании множества ресурсов одного класса:

1. Задаем начальные условия: A — вектор доступных ресурсов, представляющий собой множество маркированных позиций; R — матрица запросов, представляет собой множество переходов выходящих из позиций; C — матрица текущего распределения, представляется множеством переходов входящие в позиции (места); L — пустой список, множество обработанных переходов.

2. Ищем немаркированный переход T_i, для которого i-я строка матрицы R меньше вектора A или равна ему, т. е. данный переход активен.

3. Если такой переход найден, прибавляем i-ю строку матрицы C к вектору A, а так же отнимаем i-ю строку матрицы R от вектора A и маркируем переход добавляя его в список L, т. е. определяем новую разметку сети, возвращаемся к шагу 2.

4. Если таких переходов не существует, то переходим к шагу 5

5. Если остались не маркированные переходы, то данные переходы попали в

тупик.

Заключение

Так как на данный момент не существует метода анализа сети Петри дающего достаточные условия для обнаружения тупика в сети, предложен новый метод анализа сети Петри для обнаружения тупиков в сети возникающих при разделении ресурсов. Метод показывает достаточность возникновения тупика, необходимым условием для этого является безопасность сети.

Список литературы

[1] Котов В.Е. Сети Петри. М.: Наука. 1984. 160 с.

[2] Советов Б.Я., Яковлев С.А. Моделирование систем. М.: Высшая школа. 2005. 344 с.

[3] Таненбаум Э. Современные операционные системы. Спб:Питер. 2011. 1120 с.

[4] Рудаков И. В., Ребриков А. В. Масштабирование алгоритмов для автоматической генерации модульных тестов // Вестник МГТУ им. Н. Э. Баумана. Сер. Приборостроение. 2011. № 4. С. 119-124.

[5] Рудаков И. В., Ребриков А. В. Проверка выполнения функциональных требований к алгоритму на основе структурной генерации модульных тестов // Вестник МГТУ им. Н. Э. Баумана. Сер. Приборостроение. 2012. Спец. вып. 2: Программная инженерия. С. 67-79.